

Linguagem XPath (XML Path Language)

Baseado nos slides do professor Paulo Trigo
Todas as alterações são da responsabilidade do professor António Teófilo
e do professor Diogo Remédios

O que é o XPath?

- É uma linguagem para descrever caminhos em documentos XML
 - a linguagem XPath tem uma sintaxe própria (diferente do XML)
- Na perspectiva XPath, o documento XML representa uma árvore
 - no qual se podem traçar caminhos
- ... uma expressão XPath representa um caminho numa árvore
 - permite “navegar” entre elementos e atributos (num documento XML)
- As expressões XPath são usadas noutras linguagens
 - e.g. XSLT (“eXtensible Stylesheet Language Transformations”)
 - e.g. XQuery
 - e.g. XSD (XML Schema Definition)
 - e.g. navegação em DOM
 - ... conhecer XPath é requisito para uma utilização avançada do XML!

Novamente a lista de bares em XML

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE lista_bar_cerveja [
  <!ELEMENT lista_bar_cerveja (bar+, cerveja+, bar_cerveja*)>

  <!ELEMENT bar EMPTY>
  <!ATTLIST bar oBar ID #REQUIRED>
  <!ATTLIST bar nome CDATA #REQUIRED>

  <!ELEMENT cerveja EMPTY>
  <!ATTLIST cerveja aCerveja ID #REQUIRED>
  <!ATTLIST cerveja nome CDATA #REQUIRED>

  <!ELEMENT bar_cerveja (#PCDATA)>
  <!ATTLIST bar_cerveja oBar IDREF #REQUIRED>
  <!ATTLIST bar_cerveja aCerveja IDREF #IMPLIED >
]>
```

ListaBares.xml

```
<lista_bar_cerveja>

  <bar oBar="b1" nome="O Bar Inglês"/>
  <bar oBar="b2" nome="O Pinguim"/>

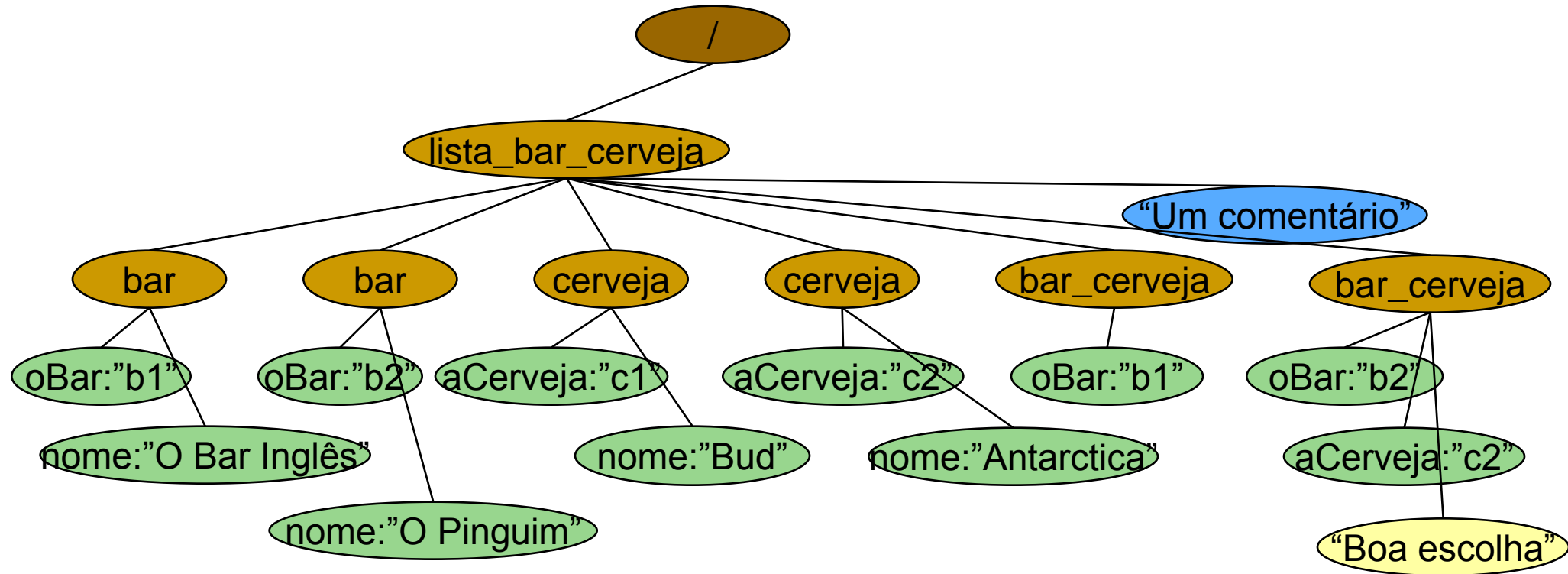
  <cerveja aCerveja="c1" nome="Bud"/>
  <cerveja aCerveja="c2" nome="Antarctica"/>

  <bar_cerveja oBar="b1"/>

  <bar_cerveja oBar="b2" aCerveja="c2" >Boa Escolha!
</bar_cerveja>
  <!-- Um comentário -->

</lista_bar_cerveja>
```

A lista de bares em formato Árvore



- Raiz do documento, é um elemento
- Elemento
- Atributo

- texto
- Comentário

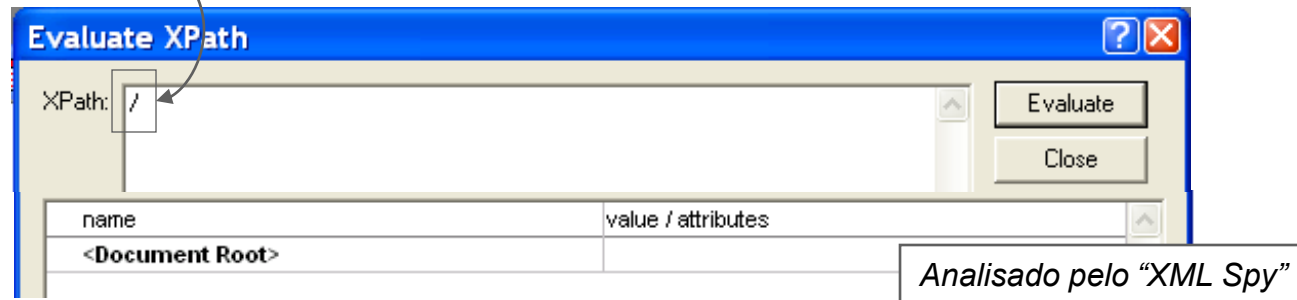
XPath: a sua perspectiva sobre um documento XML

- A linguagem XPath considera que um documento XML
 - é uma árvore de nós
 - ... uma expressão XPath devolve uma lista de nós
- O XPath considera 7 tipos de nós

Tipo de nó	Expressões (que devolvem esse tipo de nó)
raiz	/
elemento	bar, element()
atributo	@oBar, attribute()
texto	text()
comentário	comment()
instrução de processamento	processing-instruction()
espaço de nomes	namespace-uri(), local-name(), name()

Passo de localização (“location step”)

- Um passo de localização (“location step”) pode ser:
 - ❑ o símbolo `/` (que representa a raiz do documento), ou `//` (dentro de)
 - ❑ o nome de um elemento (e.g. `bar`), ou `element()`
 - ❑ o nome de um atributo precedido pelo símbolo `@` (e.g. `@oBar`), ou `attribute()`
 - ❑ a expressão `text()`
 - ❑ a expressão `comment()`
 - ❑ a expressão `processing-instruction()`
- O símbolo `/` representa a raiz do documento e designa-se
 - ❑ nó raiz (“root node”), ou nó documento (“document node”),
 - ❑ ou raiz do documento (“document root”)



Caminho de localização (“location path”)

- Um caminho de localização (“location path”)
 - é uma expressão XPath
- Um caminho de localização é constituído por
 - passos de localização (“location steps”)
- Diversos passos de localização (“location steps”) são combinados
 - pelo símbolo / (ou por //)
 - e.g. `/lista_bar_cerveja/bar_cerveja/text()`
 - ... idêntico à definição de um caminho no sistema de ficheiros Linux!
- Cada passo de localização é avaliado
 - em relação a um nó de contexto (“context node”)
 - e.g. em `/lista_bar_cerveja/bar_cerveja/text()`
 - ... o nó de contexto de `lista_bar_cerveja` é: `/`

... outros símbolos num caminho de localização

Símbolo	Descrição
<i>// expressão</i>	selecciona os nós que, a partir do nó corrente, verificam a <i>expressão</i> independentemente da localização desses nós
.	selecciona o nó corrente
..	selecciona o nó ascendente do corrente
*	selecciona todos os nós do tipo elemento
@*	selecciona todos os nós do tipo atributo
node()	selecciona todos os nós de todos os tipos
<i>expr₁ expr₂ . . . expr_n</i>	selecciona os nós que verificam uma das <i>expr_i</i>

node() – no XMLSpy não selecciona os nós do tipo atributo (erro face à especificação: <http://www.w3.org/TR/xpath>)

O caminho de localização e o seu valor

- Quando aplicado a um documento
 - um caminho de localização, tem um valor
- O valor de um caminho de localização é
 - uma lista de componentes ... note-se que não é um conjunto
 - ... pode ter componentes repetidos e existe relação de ordem!
- Cada componente pode ser:
 - um nó, e.g. nome="Bud" (nó atributo), ou
 - um valor atómico, e.g. "Bud"
- Em geral e como regra simples pode considerar-se que
 - um nó é uma marca ("tag") e tudo o que ela contém
 - um valor atómico é um nó sem descendente
 - ... o valor atómico é uma folha na hierarquia do documento!

O caminho de localização e o seu valor: exemplos

/bar															
//bar	<table><thead><tr><th>name</th><th>value / attributes</th></tr></thead><tbody><tr><td>bar</td><td>oBar="b1" nome="O Bar Inglês"</td></tr><tr><td>bar</td><td>oBar="b2" nome="O Pinguim"</td></tr></tbody></table>	name	value / attributes	bar	oBar="b1" nome="O Bar Inglês"	bar	oBar="b2" nome="O Pinguim"								
name	value / attributes														
bar	oBar="b1" nome="O Bar Inglês"														
bar	oBar="b2" nome="O Pinguim"														
//@oBar	<table><thead><tr><th>name</th><th>value / attributes</th></tr></thead><tbody><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr></tbody></table>	name	value / attributes	= oBar	b1	= oBar	b2	= oBar	b1	= oBar	b2				
name	value / attributes														
= oBar	b1														
= oBar	b2														
= oBar	b1														
= oBar	b2														
//bar //@oBar	<table><thead><tr><th>name</th><th>value / attributes</th></tr></thead><tbody><tr><td>bar</td><td>oBar="b1" nome="O Bar Inglês"</td></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>bar</td><td>oBar="b2" nome="O Pinguim"</td></tr><tr><td>= oBar</td><td>b2</td></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr></tbody></table>	name	value / attributes	bar	oBar="b1" nome="O Bar Inglês"	= oBar	b1	bar	oBar="b2" nome="O Pinguim"	= oBar	b2	= oBar	b1	= oBar	b2
name	value / attributes														
bar	oBar="b1" nome="O Bar Inglês"														
= oBar	b1														
bar	oBar="b2" nome="O Pinguim"														
= oBar	b2														
= oBar	b1														
= oBar	b2														
//@oBar/../../aCerveja	<table><thead><tr><th>name</th><th>value / attributes</th></tr></thead><tbody><tr><td>= aCerveja</td><td>c2</td></tr></tbody></table>	name	value / attributes	= aCerveja	c2										
name	value / attributes														
= aCerveja	c2														
/lista_bar_cerveja//text()	<table><thead><tr><th>name</th><th>value / attributes</th></tr></thead><tbody><tr><td>Abc</td><td>Boa Escolha!</td></tr></tbody></table>	name	value / attributes	Abc	Boa Escolha!										
name	value / attributes														
Abc	Boa Escolha!														

... o caminho de localização e o seu valor: exemplos

<code>//bar/@oBar</code> <code>//bar_cerveja/@oBar</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr></table>	name	value / attributes	= oBar	b1	= oBar	b2	= oBar	b1	= oBar	b2
name	value / attributes										
= oBar	b1										
= oBar	b2										
= oBar	b1										
= oBar	b2										
<code>//bar/@oBar/..</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td><> bar</td><td>oBar="b1" nome="O Bar Inglês"</td></tr><tr><td><> bar</td><td>oBar="b2" nome="O Pinguim"</td></tr></table>	name	value / attributes	<> bar	oBar="b1" nome="O Bar Inglês"	<> bar	oBar="b2" nome="O Pinguim"				
name	value / attributes										
<> bar	oBar="b1" nome="O Bar Inglês"										
<> bar	oBar="b2" nome="O Pinguim"										
<code>//bar/@oBar/.</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td>= oBar</td><td>b1</td></tr><tr><td>= oBar</td><td>b2</td></tr></table>	name	value / attributes	= oBar	b1	= oBar	b2				
name	value / attributes										
= oBar	b1										
= oBar	b2										
<code>/lista_bar_cerveja/bar</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td><> bar</td><td>oBar="b1" nome="O Bar Inglês"</td></tr><tr><td><> bar</td><td>oBar="b2" nome="O Pinguim"</td></tr></table>	name	value / attributes	<> bar	oBar="b1" nome="O Bar Inglês"	<> bar	oBar="b2" nome="O Pinguim"				
name	value / attributes										
<> bar	oBar="b1" nome="O Bar Inglês"										
<> bar	oBar="b2" nome="O Pinguim"										
<code>//comment()</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td><></td><td>Um comentário</td></tr><tr><td><></td><td>Outro comentário</td></tr></table>	name	value / attributes	<>	Um comentário	<>	Outro comentário				
name	value / attributes										
<>	Um comentário										
<>	Outro comentário										
<code>/lista_bar_cerveja</code> <code>/comment()</code>	<table><tr><th>name</th><th>value / attributes</th></tr><tr><td><></td><td>Outro comentário</td></tr></table>	name	value / attributes	<>	Outro comentário						
name	value / attributes										
<>	Outro comentário										

... o meta-carácter *: exemplos

/lista_bar_cerveja/*

Todos os
elementos de
/lista_bar_cerveja

name	value / attributes
<> bar	oBar="b1" nome="O Bar Inglês"
<> bar	oBar="b2" nome="O Pinguim"
<> cerveja	aCerveja="c1" nome="Bud"
<> cerveja	aCerveja="c2" nome="Antarctica"
<> bar_cerveja	oBar="b1"
<> bar_cerveja	Boa Escolha!

/lista_bar_cerveja/bar/@*

name	value / attributes
= oBar	b1
= nome	O Bar Inglês
= oBar	b2
= nome	O Pinguim

/*/bar/*

elementos
descendentes de
bar (de 2º nível)

name	value / attributes
------	--------------------

/*/ */@*

atributos no
3º nível da
hierarquia

name	value / attributes
= oBar	b1
= nome	O Bar Inglês
= oBar	b2
= nome	O Pinguim
= aCerveja	c1
= nome	Bud
= aCerveja	c2
= nome	Antarctica
= oBar	b1
= aCerveja	
= oBar	b2
= aCerveja	c2

O nó do tipo espaço de nomes “namespace”

- O espaço de nomes (“namespace”) é um tipo de nó, pelo que
 - pode ser referido numa expressão XPath
 - ... existem funções especializadas para operar sobre espaços de nomes
- As funções que operam sobre um espaço de nomes
 - aceitam todas, como argumento, uma qualquer expressão XPath
 - ... e.g. `namespace-uri(//ultNome)`
- O valor de uma expressão XPath é uma lista de nós, pelo que
 - as funções sobre “namespaces” consideram o primeiro nó da lista
- As 3 funções sobre “namespaces”, para uma expressão `expr`, são:
 - `namespace-uri(expr)` \equiv nome do URI do primeiro nó em `expr`
 - `local-name(expr)` \equiv parte local do nome do primeiro nó em `expr`
 - `name(expr)` \equiv nome qualificado do primeiro nó em `expr`

O nó do tipo espaço de nomes: exemplo

XML (exemplo):

```
<emp:secretaria xmlns:emp="http://Empregados">
  <emp:nrBI>1230321</emp:nrBI>
  <nomeCompleto>
    <emp:primNome>Maria</emp:primNome>
    <ultNome>Airam</ultNome>
  </nomeCompleto>
</emp:secretaria>
```

... este nome está no
“namespace”
http://Empregados?

	Sim	Não
secretaria	✓	
nrBI	✓	
nomeCompleto		✓
primNome	✓	
ultNome		✓

nomes locais: não estão em nenhum “namespace”

namespace-uri (/emp:secretaria)

name	value / attributes
xs:string	http://Empregados

namespace-uri (//ultNome)

name	value / attributes
xs:string	

name (/emp:secretaria) -> emp:secretaria

local-name (/emp:secretaria) -> secretaria

Os namespaces para serem reconhecidos assim têm de estar declarados no elemento raiz

... “namespace”: exemplo

... em que “namespace” estou?

```
<emp:secretaria xmlns:emp="http://Empregados"
                 xmlns="http://EmpOmissao">
  <emp:nrBI>1230321</emp:nrBI>
  <nomeCompleto grupo="A" emp:estado="S">
    <emp:primNome>Maria</emp:primNome>
    <ultNome>Airam</ultNome>
  </nomeCompleto>
</emp:secretaria>
```

	emp	omissão	local
secretaria	✓		
nrBI	✓		
grupo			✓
estado	✓		
nomeCompleto		✓	
primNome	✓		
ultNome		✓	

recordar: atributos não qualificados são locais

namespace-uri (/emp:secretaria)	<table><tr><td>name</td><td>value / attributes</td></tr><tr><td>xs:string</td><td>http://Empregados</td></tr></table>	name	value / attributes	xs:string	http://Empregados
name	value / attributes				
xs:string	http://Empregados				
namespace-uri (//ultNome)	<table><tr><td>name</td><td>value / attributes</td></tr><tr><td>xs:string</td><td>http://EmpOmissao</td></tr></table>	name	value / attributes	xs:string	http://EmpOmissao
name	value / attributes				
xs:string	http://EmpOmissao				
namespace-uri (//@grupo)	<table><tr><td>name</td><td>value / attributes</td></tr><tr><td>xs:string</td><td></td></tr></table>	name	value / attributes	xs:string	
name	value / attributes				
xs:string					
//@grupo	<table><tr><td>name</td><td>value / attributes</td></tr><tr><td>= grupo</td><td>A</td></tr></table>	name	value / attributes	= grupo	A
name	value / attributes				
= grupo	A				

Predicado

- Um predicado é aplicado a um passo de localização (“location step”)
 - sobre o qual se define uma condição
 - ... um predicado é uma expressão XPath (tal como o “location path”)
- Num predicado, a condição é escrita entre parêntesis rectos [e]
 - e.g. elemento `bar` com atributo `nome` com valor ‘O Pinguim’
 - `//bar[@nome='O Pinguim']`
- Os operadores disponíveis para escrever a condição
 - são: `=`, `!=`, `>`, `<`, `>=`, `<=`, `and`, `or`
 - os valores podem escrever-se entre plicas ou aspas
 - os parêntesis curvos podem usar-se para explicitar precedências
 - ... `//bar[@nome='O Pinguim' or //@acerveja="Bud"]`
- Mais funcionalidades
 - `/bookstore/book[1]` → devolve o primeiro elemento *book*
 - `/bookstore/book[last()]` → devolve o último elemento *book*
 - `/bookstore/book[last()-1]` → devolve o penúltimo
 - `/bookstore/book[position()< 3]` → devolve os primeiro dois
 - `//title[@lang]` → devolve os elementos *title* que tenham o atributo *lang*

Recordar a lista de bares em XML

ListaBares.xml

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE lista_bar_cerveja [
  <!ELEMENT lista_bar_cerveja (bar+, cerveja+, bar_cerveja*)>

  <!ELEMENT bar EMPTY>
  <!ATTLIST bar oBar ID #REQUIRED>
  <!ATTLIST bar nome CDATA #REQUIRED>

  <!ELEMENT cerveja EMPTY>
  <!ATTLIST cerveja aCerveja ID #REQUIRED>
  <!ATTLIST cerveja nome CDATA #REQUIRED>

  <!ELEMENT bar_cerveja (#PCDATA)>
  <!ATTLIST bar_cerveja oBar IDREF #REQUIRED>
  <!ATTLIST bar_cerveja aCerveja IDREF #IMPLIED>
]>
```

```
<lista_bar_cerveja>

  <bar oBar="b1" nome="O Bar Inglês"/>
  <bar oBar="b2" nome="O Pinguim"/>

  <cerveja aCerveja="c1" nome="Bud"/>
  <cerveja aCerveja="c2" nome="Antarctica"/>

  <bar_cerveja oBar="b1"/>

  <bar_cerveja oBar="b2" aCerveja="c2" >Boa Escolha!
</bar_cerveja>
  <!-- Outro comentário -->

</lista_bar_cerveja>
```

Predicados: exemplos


elemento `bar` com atributo de nome 'O Pinguim'

```
//bar[@nome='O Pinguim']
```

name	value / attributes
 bar	oBar="b2" nome="O Pinguim"


atributo `oBar` do `bar` com nome 'O Pinguim'

```
//bar[@nome='O Pinguim']/@oBar
```

name	value / attributes
 oBar	b2

elemento `bar_cerveja` que refere o `bar` com nome 'O Pinguim'

```
//bar_cerveja[@oBar=//bar[@nome='O Pinguim']/@oBar]
```


name	value / attributes
 bar_cerveja	Boa Escolha!

```
//bar_cerveja[@oBar=//bar/@oBar] -> elems: bar_cerveja b1 e bar_cerveja b2
```

... outros predicados: exemplos

elemento bar de nome 'O Pinguim' ou com aCerveja 'Bud'
(bar não tem atributo aCerveja mas a expressão é correcta)

```
//bar[@nome='O Pinguim' or //@aCerveja="Bud"]
```

name	value / attributes
 bar	oBar="b2" nome="O Pinguim"


qualquer elemento com atributo de nome 'Bud'

```
//*[@nome='Bud']
```

name	value / attributes
 cerveja	aCerveja="c1" nome="Bud"

o atributo aCerveja de qualquer elemento com atributo de nome 'Bud'

```
//*[@nome='Bud']/@aCerveja
```

name	value / attributes
 aCerveja	c1

`//*[@nome and @nome2]` → elementos que tenham o atributo nome e o atributo nome2

`count(//bar_cerveja[@oBar="b2"]/@*)` → nº de atributos dentro dos elementos bar_cerveja que tem atributo oBar="b2"

Localizar nomes de diferentes namespaces

```
<empregados>
  <emp:secretaria xmlns:emp="http://Empregados">
    <emp:nrBI>1230321</emp:nrBI>
    <nomeCompleto>
      <emp:primNome>Maria</emp:primNome>
      <ultNome>Airam</ultNome>
    </nomeCompleto>
  </emp:secretaria>
  <emp:secretaria xmlns:emp="http://Empregados2">
    <emp:nrBI>1230321</emp:nrBI>
    <nomeCompleto>
      <emp:primNome>Manuela</emp:primNome>
      <ultNome>Airam</ultNome>
    </nomeCompleto>
  </emp:secretaria>
</empregados>
```

Localização dos elementos **secretaria** do namespace “**http://Empregados**”

```
/empregados/*[namespace-uri()='http://Empregados' and local-name()='secretaria']
```

Localização dos elementos **secretaria** do namespace “**http://Empregados2**”



```
/empregados/*[namespace-uri()='http://Empregados2' and local-name()='secretaria']
```

... desigualdades e . (nó corrente): exemplos

```
<listaDePrecos>
  <cerveja nome="Bud" preco="2.50"/>
  <cerveja nome="Miller" preco="3.00"/>
  <cerveja nome="Antarctica" preco="2.00"/>
  <outraBebida>A<preco>1.00</preco></outraBebida>
  <outraBebida>B<preco>1.50</preco></outraBebida>
  <outraBebida>C<preco>5.00</preco></outraBebida>
</listaDePrecos>
```

cervejas até 2.50 (inclusive)

//cerveja[@preco <= 2.50]



name	value / attributes
 cerveja	nome="Bud" preco="2.50"
 cerveja	nome="Antarctica" preco="2.00"

//cerveja/@preco[. <= 2.50]

name	value / attributes
= preco	2.50
= preco	2.00

outras bebidas até 2.50 inclusive)



//outraBebida[preco <= 2.50]

name	value / attributes
 outraBebida	A
 outraBebida	B

a entidade tem texto!

//outraBebida[preco <= 2.50]/*

//outraBebida/preco[. <= 2.50]

name	value / attributes
 preco	1.00
 preco	1.50

Noção de “eixo”

- Cada passo de localização (“location path”) define um percurso
 - ao longo de um eixo a partir de um nó de contexto (ou nó corrente)
- O nó de contexto é um dos 7 tipos considerados na linguagem
 - na prática, é comum ser o nó raiz ou um nó elemento
- O nó de contexto (seja ele qual for) está sempre relacionado
 - com qualquer outro nó que esteja presente no documento
- As possíveis relações entre quaisquer dois nós
 - representam eixos nos quais se pode percorrer um documento
 - ... ou formas de separar os nós em subconjuntos não disjuntos
 - ... começando num qualquer nó de contexto
 - e.g. // @oBar / .. / @aCerveja
 - ... atributo oBar, passar para ascendente e atributo aCerveja

Os 13 eixos

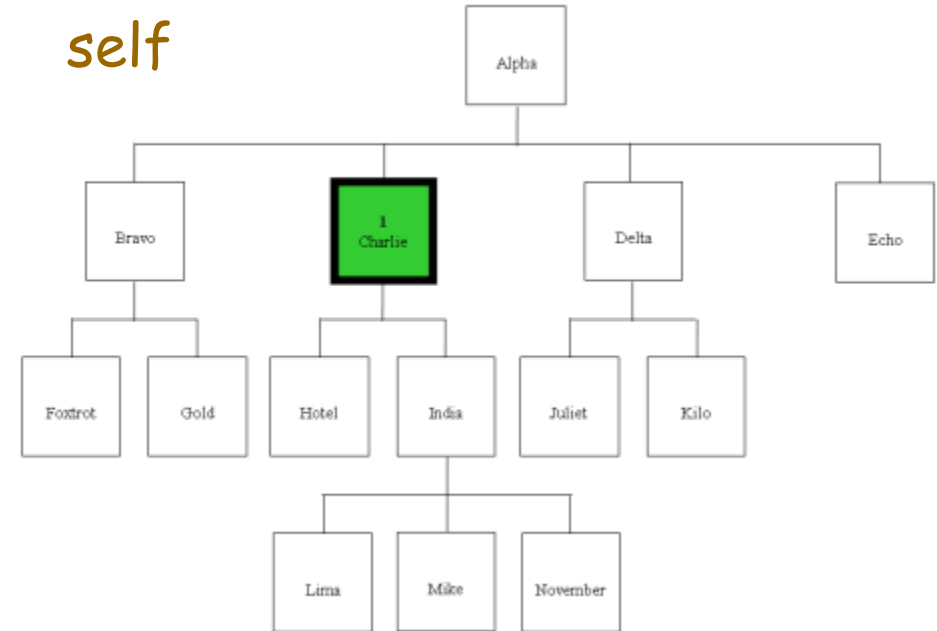
- Foram definidos 13 eixos (formas de relação entre quaisquer 2 nós)
 - o nome do eixo é separado por :: do nó que o segue, e.g. /child::*

Nome do Eixo	Significado
self	o nó corrente
child	nós filho (eixo de omissão)
descendant	toda a descendência; nós filho, nós filho do filho, etc
descendant-or-self	o nó corrente e toda a sua descendência
parent	o nó que contém o corrente; o nó raiz não tem parent
ancestor	todos os antepassados do nó corrente; o nó raiz não tem ancestor
ancestor-or-self	o nó corrente e todos os seus antepassados
following	nós irmãos escritos depois do nó corrente, e seus descendentes
preceding	nós irmãos escritos antes do nó corrente, e seus descendentes
following-sibling	nós irmãos escritos depois do nó corrente
preceding-sibling	nós irmãos escritos antes do nó corrente
attribute	todos os atributos do nó corrente
namespace	todos os “namespace” no âmbito do nó corrente

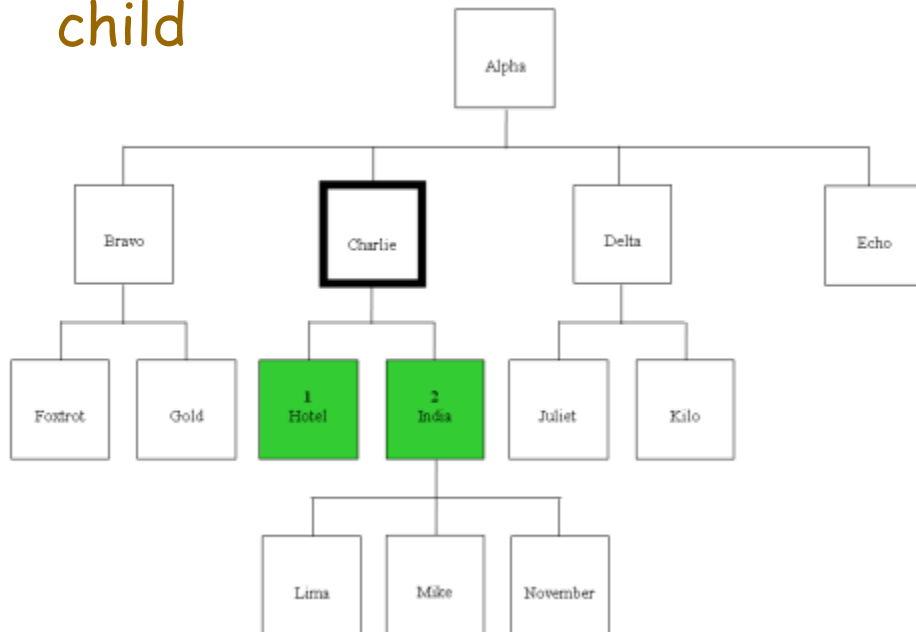
Eixos

A negrito assinala-se o nó corrente.
A verde os nós devolvidos

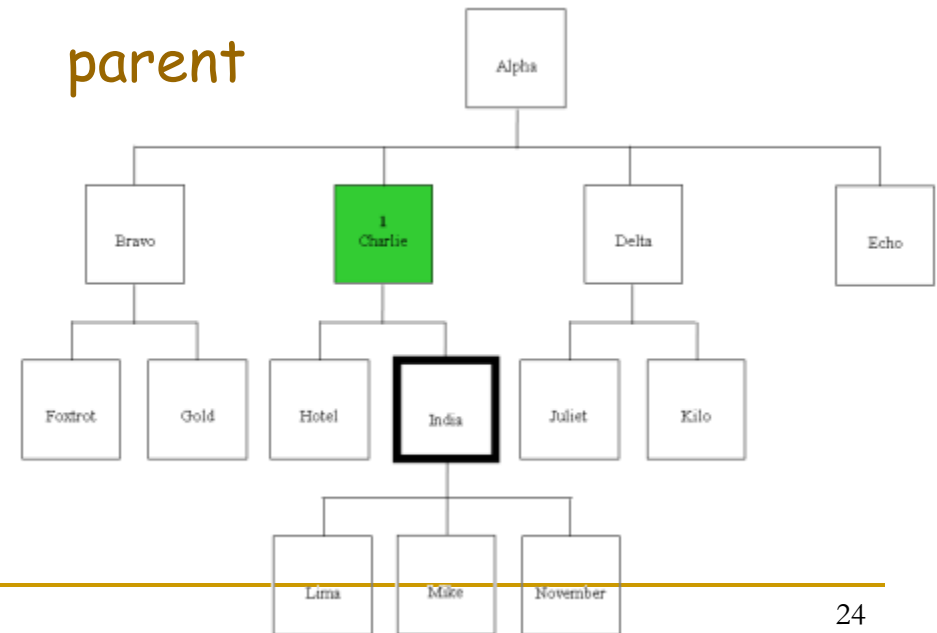
self



child

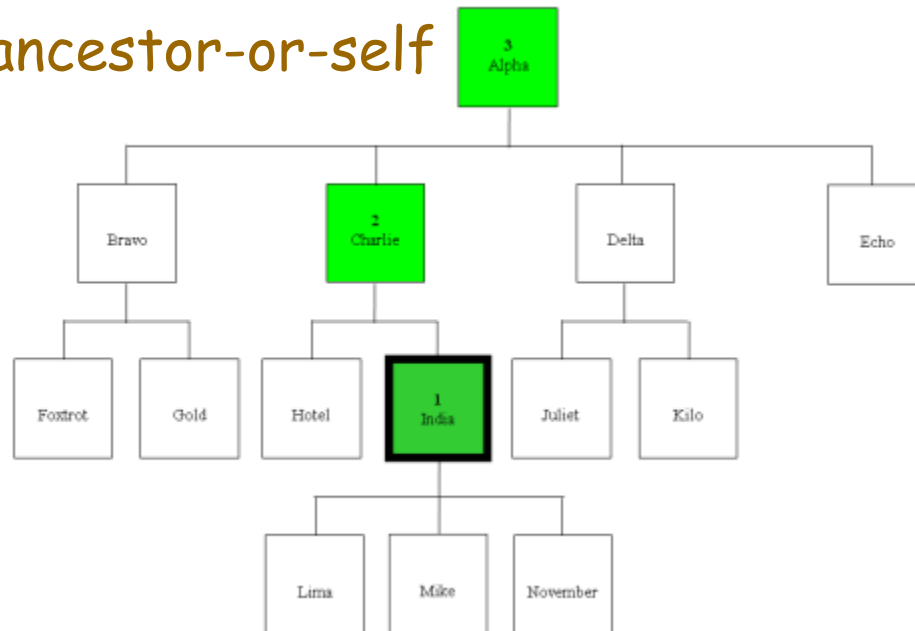
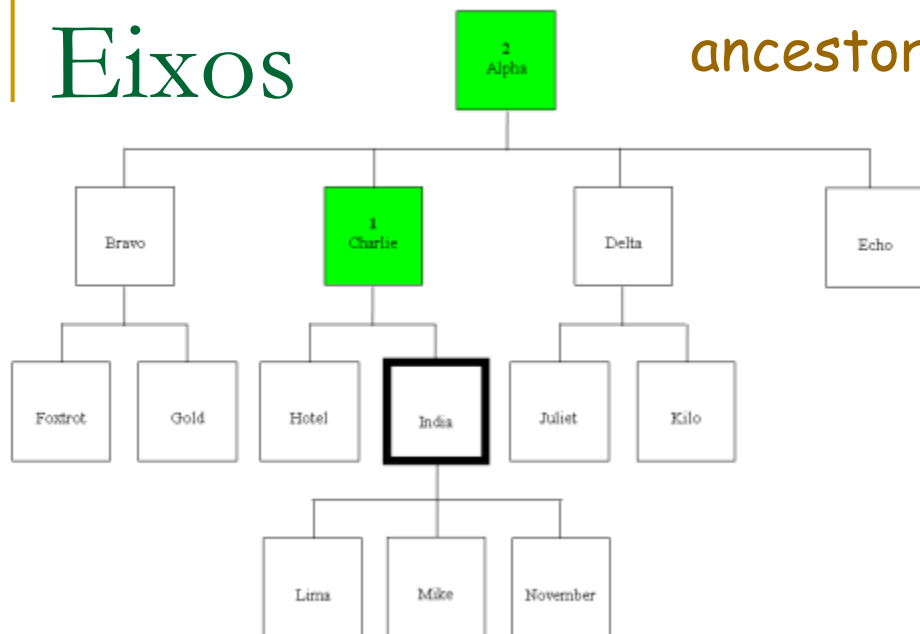


parent



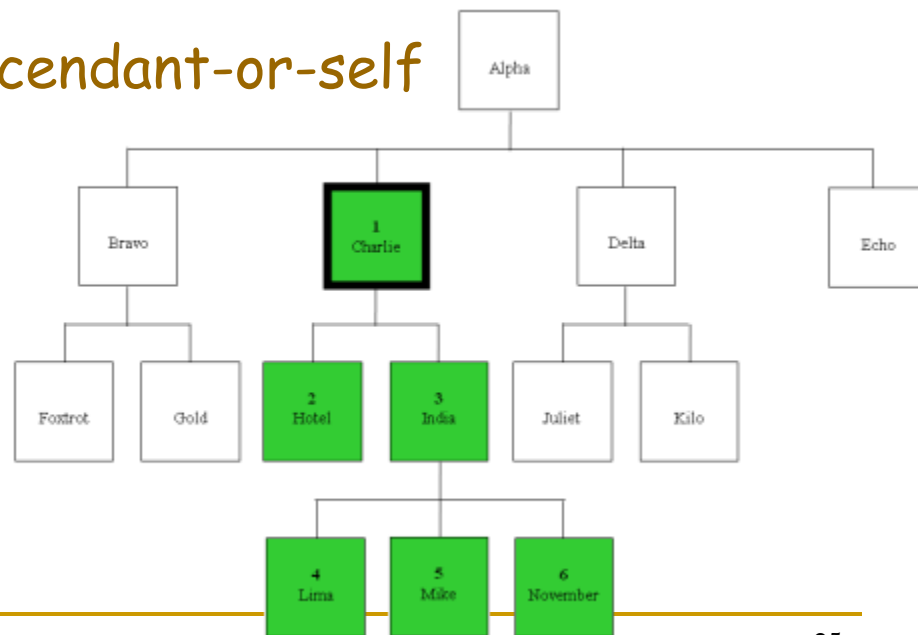
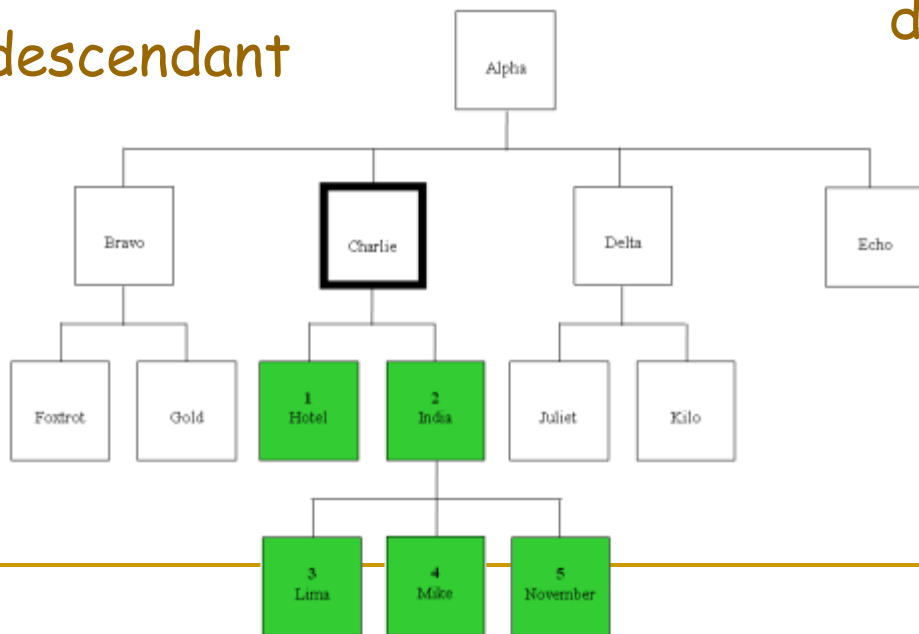
Eixos

ancestor ancestor-or-self



descendant

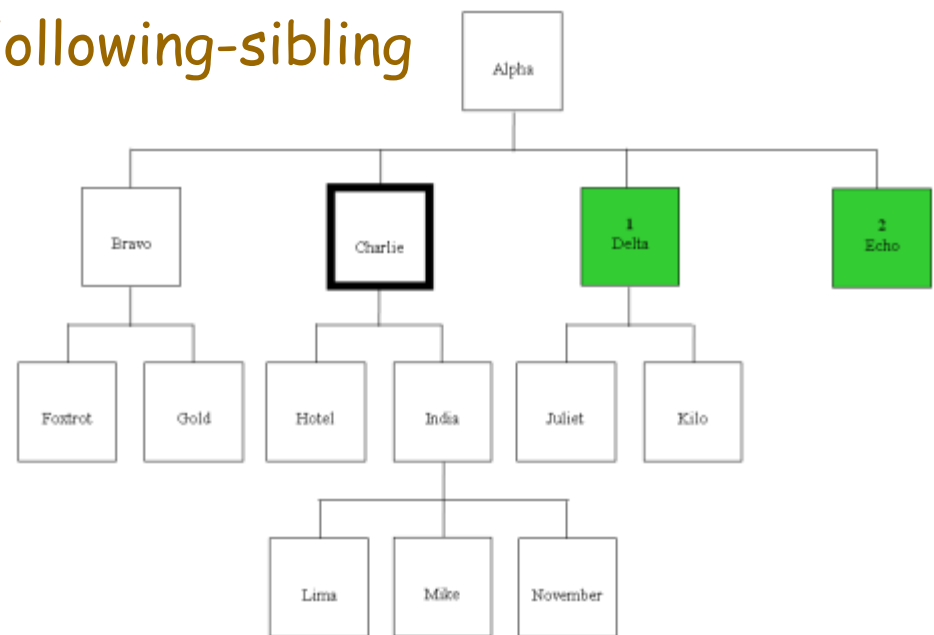
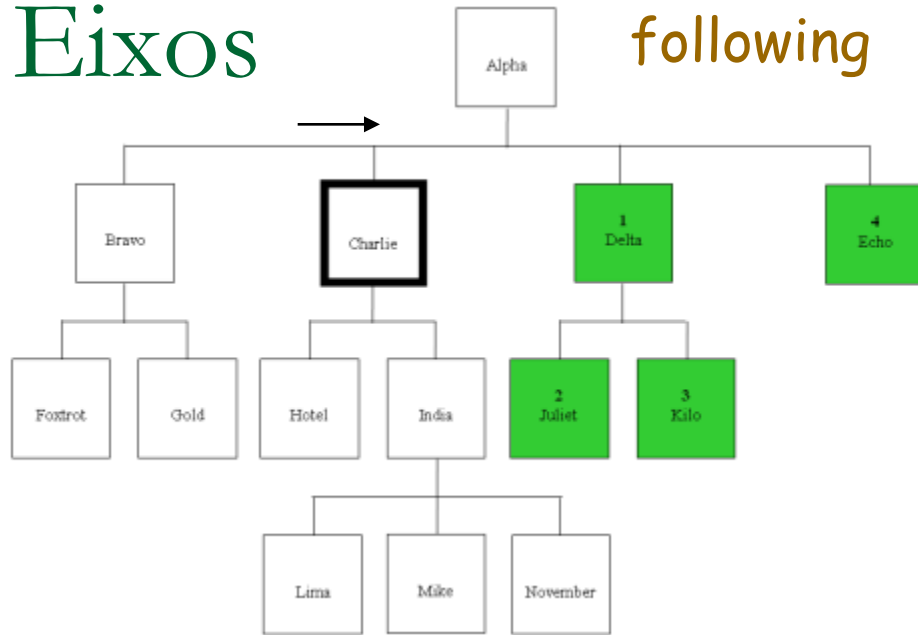
descendant-or-self



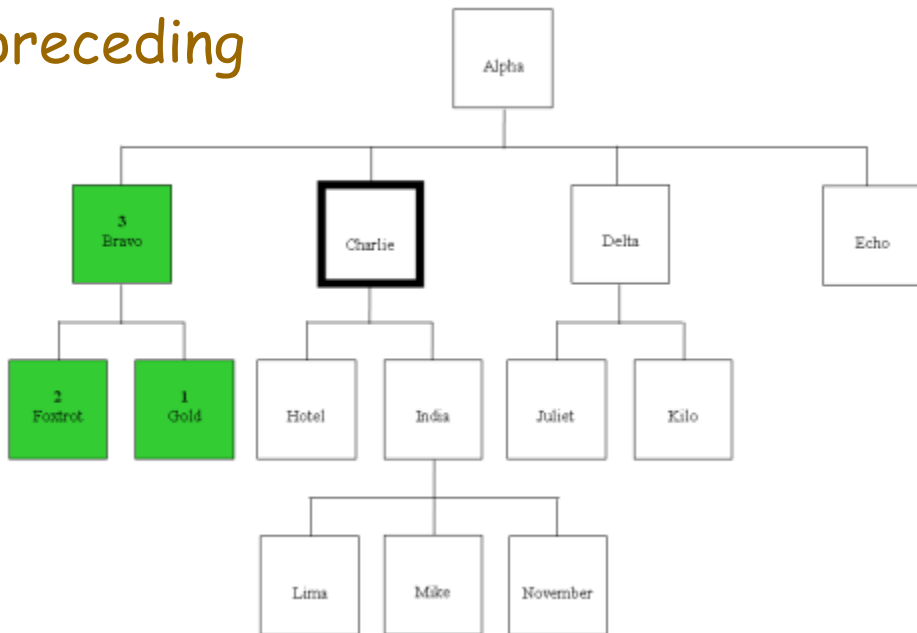
Eixos

following

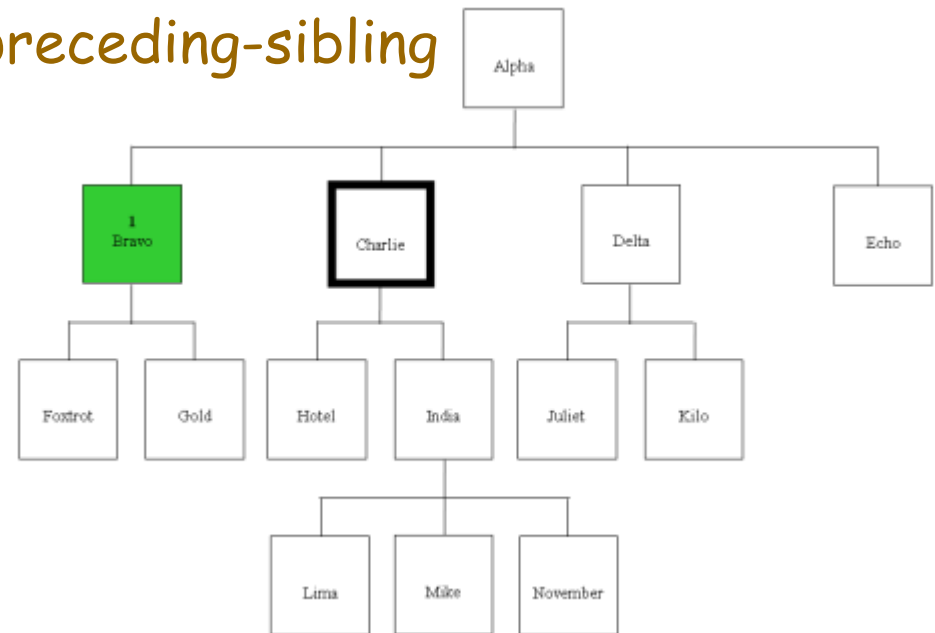
following-sibling



preceding



preceding-sibling



... eixos: formas abreviadas de indicação

- Em cada passo de localização pode seguir-se por um dos 13 eixos
 - ... o eixo de omissão é `child` (todos os filhos do nó corrente)
 - e.g. `/listaDePrecos/cerveja` é a forma abreviada de escrever
 - `/listaDePrecos/child::cerveja`
- Outras formas abreviadas (geralmente usadas)

Forma abreviada	Forma completa (eixo explícito)
<code>//</code>	<code>/descendant-or-self::node() /</code>
<code>.</code>	<code>self::node()</code>
<code>..</code>	<code>parent::node()</code>
<code>@<i>nome</i></code>	<code>attribute::<i>nome</i></code>

Exemplo:



Explorar a relação de ordem do texto no documento






Importante: **existe relação de ordem num conteúdo XML***

Uma expressão XPath pode usar essa relação de ordem
(recordar: o valor de uma expressão XPath é uma lista de nós)

Quais os nós que
estão depois da
outraBebida B?

```
<listaDePrecos>
  <outraBebida>A<preco>1.00</preco></outraBebida>
  <outraBebida>B<preco>1.50</preco></outraBebida>
  <outraBebida>C<preco>5.00</preco></outraBebida>
  <cerveja nome="Bud" preco="2.50"/>
  <cerveja nome="Miller" preco="3.00"/>
  <cerveja nome="Antarctica" preco="2.00"/>
</listaDePrecos>
```

`/listaDePrecos/outraBebida[text()='B']/following::*`

name	value / attributes
 outraBebida	C
 preco	5.00
 cerveja	nome="Bud" preco="2.50"
 cerveja	nome="Miller" preco="3.00"
 cerveja	nome="Antarctica" preco="2.00"

* Na perspectiva XPath

Em síntese: uma expressão XPath pode ser...

- ... um passo de localização (“location step”)
 - a expressão mais simples
 - e.g. /
- ... um caminho de localização (“location path”)
 - sequência de passos de localização separados por /
 - e.g. /listaDePrecos/outraBebida
- ... um predicado (“predicate”)
 - um caminho de localização com condições escritas entre [e]
 - e.g. /listaDePrecos/outraBebida[text()='B']
- Adicionalmente uma expressão XPath pode ainda ter
 - indicação explícita de eixos
 - e.g. /listaDePrecos/cerveja/**preceding**::outraBebida
 - evocação de funções XPath (descritas em seguida...)

Avaliar expressão XPath em linguagem de “scripting”

- O objecto `XMLDOM` (da Microsoft) permite
 - ❑ carregar um documento XML, e
 - ❑ o método `selectNodes()` avalia uma expressão XPath, e
 - ❑ devolve a lista de nós que satisfazem essa expressão
- O objecto `XMLDOM` pode ser instanciado usando “vbscript”
 - ❑ ... e assim avaliar a expressão XPath e obter a lista de nós resultante

```
set xmlDoc = CreateObject("Microsoft.XMLDOM")
xmlDoc.async = "false"
xmlDoc.load("nomeFicheiro.xml")

set nodes = xmlDoc.selectNodes(expressãoXPath)
```

- Com a lista resultante da avaliação (`nodes` no exemplo atrás)
 - ❑ pode gerar-se um novo documento que apenas contém essa lista!

Exemplo: “avaliar expressão XPath com XMLDOM”

```
<html>
<body>
<script type="text/vbscript">

set xmlDoc=CreateObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("listaDePrecos.xml")

set nodes = xmlDoc.selectNodes("//cerveja[@preco <= 2.50]")

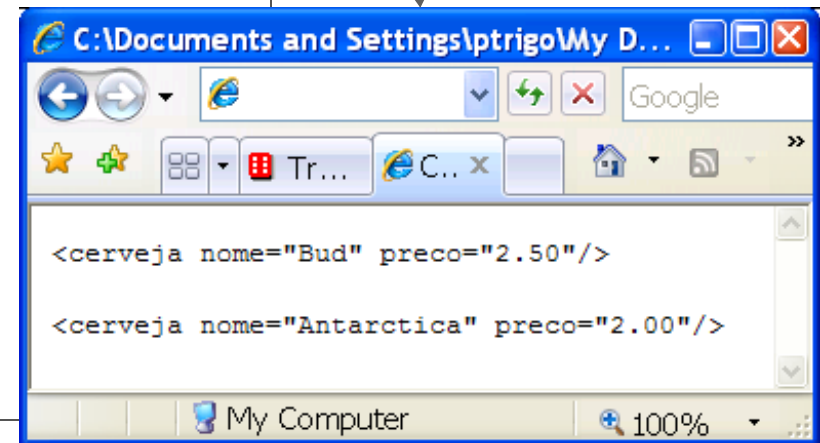
for each x in nodes
    document.write("<xmp>")
    document.write(x.xml)
    document.write("</xmp>")
next

</script>
</body>
</html>
```

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<listaDePrecos>
    <outraBebida>A<preco>1.00</preco></outraBebida>
    <outraBebida>B<preco>1.50</preco></outraBebida>
    <outraBebida>C<preco>5.00</preco></outraBebida>
    <cerveja nome="Bud" preco="2.50"/>
    <cerveja nome="Miller" preco="3.00"/>
    <cerveja nome="Antarctica" preco="2.00"/>
</listaDePrecos>
```

lista_DePrecos.xml

gerar um
documento com
a lista resultante



Ao copiar não esquecer de se transformar todas as aspas

JAVA DOM/XPath

XPathUtils e DomUtils são código desenvolvido

```
public class DomXPathApp0 {

    public static void main(String args[]) {
        try {
            // Get the DOM Document
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse("hello2.xml");

            // teste 1: NODESET -----
            NodeList nodes = XPathUtils.getNodeList("//@*", document);
            DomUtils.printNodeList(nodes);

            // teste 2: NODE -----
            Node node = XPathUtils.getSingleNode("//display2/@str", document);
            DomUtils.printNode(node);

            // teste 3: VALUE -----
            String value = XPathUtils.getNodeValue("//display2/@str", document);
            System.out.println("Value of: " + "//display2/@str" + " -> " + value);

        } catch (Exception e) { e.printStackTrace(); }
    }
}
```


Hello2.xml

Hello2.xml

```
<?xml version="1.0" ?>
<data local="Lisbon" date="2/2/2009" temperature="29°">
  xxx
  <display1 str="helloWorld1">Hello World!</display1>
  <display2 str="helloWorld2" />
  <display3 str="helloWorld3">
  <display4></display4>
</display3>
  yyy
</data>
```

```
[0] Attribute date = 2/2/2009
[1] Attribute local = Lisbon
[2] Attribute temperature = 29°
[3] Attribute str = helloWorld
[4] Attribute str = helloWorld
[5] Attribute str = helloWorld
NodeType -> ATTRIBUTE_NODE, nodeName -> str, nodeValue -> helloWorld2
Value of: //display2/@str -> helloWorld2
```

output

JAVA DOM/XPath utils

```
public class XPathUtils {
    static XPathFactory xfactory = XPathFactory.newInstance();
    static XPath xp = xfactory.newXPath();

    // função que devolve uma lista de nós
    public static NodeList getNodeList(String xpathExpression, Object source)
        throws XPathExpressionException {
        return (NodeList) xp.evaluate(xpathExpression, source, XPathConstants.NODESET);
    }

    // função que devolve apenas um nó
    public static Node getSingleNode(String xpathExpression, Object source)
        throws XPathExpressionException {
        return (Node) xp.evaluate(xpathExpression, source, XPathConstants.NODE);
    }

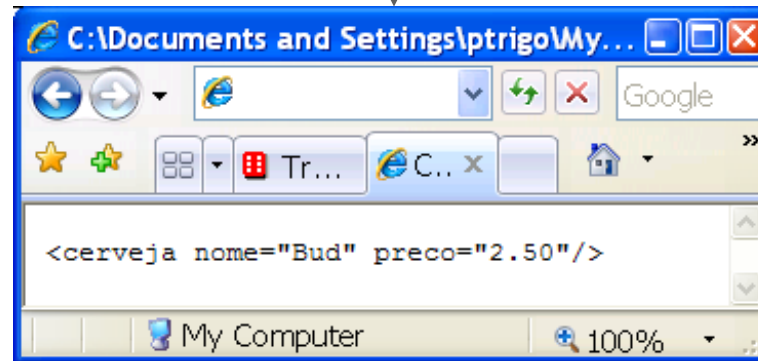
    // função que devolve o valor de um nó
    public static String getNodeValue(String xpathExpression, Object source)
        throws XPathExpressionException {
        return (String) xp.evaluate(xpathExpression, source, XPathConstants.STRING);
    }
}
```

... indexação dos nós: uma particularidade

... ver folha anterior

```
set nodes=xmlDoc.selectNodes("//cerveja[@preco <= 2.50][0]")
```

... ver folha anterior



o Internet Explorer (Microsoft) devolve o primeiro nó da lista!

Note: IE5 and later has implemented that [0] should be the first node, but **according to the W3C standard it should have been [1]!**

Informação disponível em: "http://www.w3schools.com/xpath/xpath_examples.asp"

... resolver a particularidade (i.e. seguir a norma)!

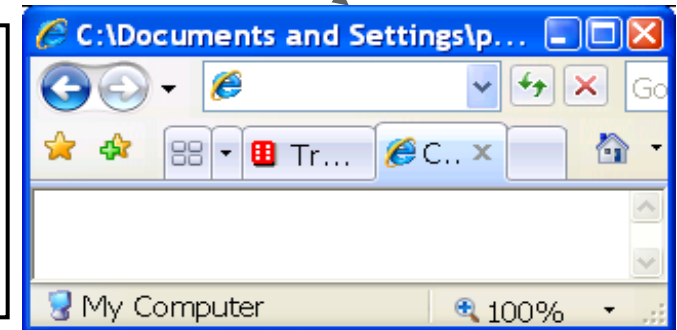
To **solve** the [0] and [1] problem in IE5+,
set the SelectionLanguage property to XPath.

... ver folha anterior

```
xmlDoc.setProperty "SelectionLanguage", "XPath"
```

```
set nodes=xmlDoc.selectNodes("//cerveja[@preco <= 2.50][0]")
```

... ver folha anterior

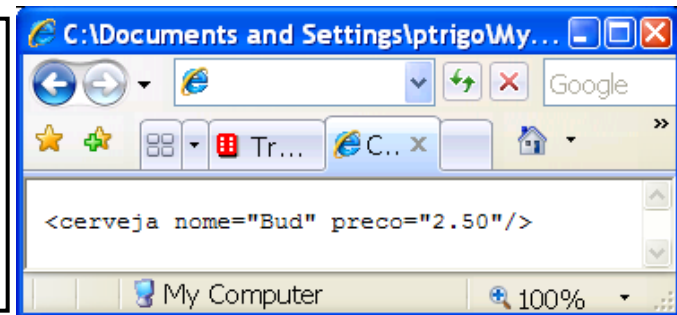


... ver folha anterior

```
xmlDoc.setProperty "SelectionLanguage", "XPath"
```

```
set nodes=xmlDoc.selectNodes("//cerveja[@preco <= 2.50][1]")
```

... ver folha anterior



Funções

- O XPath 1.0 define 27 funções para usar em expressões
 - ❑ outras tecnologias, e.g. XSLT, estendem esse conjunto de funções
 - ❑ ... o XPath 2.0 estende também o conjunto de funções do XPath 1.0
 - ❑ ... o XSLT ainda permite que o utilizador defina as suas funções!
- Cada função XPath 1.0 devolve um de 4 tipos
 - ❑ `node-set`, `string`, `boolean`, `number`
- ... assim, as funções podem organizar-se em grandes grupos
 - ❑ as que lidam com listas de nós (`node-set`)
 - ❑ as que lidam com cadeias de caracteres (`string`)
 - ❑ as que lidam com operações lógicas (`boolean`)
 - ❑ as que lidam com valores numéricos (`number`)
 - ❑ ... e outros grupos para extensões ao XPath 1.0!

Algumas funções XPath: listas de nós e “strings”

Sobre listas de nós (exploram relação de ordem do texto no documento)		
Função	Significado	Exemplo
<code>position()</code>	posição do nó corrente	<code>/listaDePrecos/cerveja[position()=1]</code>
<code>last()</code>	número do último nó	<code>/listaDePrecos/cerveja[last()]</code>
<code>count(<i>expr</i>)</code>	número de nós em <i>expr</i>	<code>count(/listaDePrecos/cerveja)</code>

Sobre cadeias de caracteres (“strings”)		
Função	Significado	Exemplo
<code>string(<i>expr</i>)</code>	converte <i>expr</i> em “string”	<code>string(//cerveja[last()]/@preco)</code>
<code>starts-with(<i>s1</i>, <i>s2</i>)</code>	<i>s1</i> começar por <i>s2</i> ?	<code>starts-with(string(//@preco), “2”)</code>
<code>contains(<i>s1</i>, <i>s2</i>)</code>	<i>s2</i> é “substring” de <i>s1</i> ?	<code>contains(string(//@preco), “2”)</code>
<code>translate(<i>s1</i>, <i>s2</i>, <i>s3</i>)</code>	<i>s3</i> substitui <i>s2</i> em <i>s1</i>	<code>translate(“aab a b”, “b ”, “a-”)</code>

cada carácter em *s2* é substituído pelo correspondente em *s3* ≡ aaa-a-a

Algumas funções XPath: “boolean” e “number”

Sobre operações lógicas (“boolean”) *		
Função	Significado	Exemplo
<code>boolean(<i>expr</i>)</code>	converte <i>expr</i> em “boolean”	<code>boolean(//cerveja[last()]/@preco=2)</code>
<code>not(<i>expr</i>)</code>	nega <i>expr</i>	<code>not(//cerveja[last()]/@preco=2.00)</code>

Sobre valores numéricos (“number”)		
Função	Significado	Exemplo
<code>number(<i>expr</i>)</code>	converte <i>expr</i> em “number”	<code>number(//cerveja[last()]/@preco)</code>
<code>sum(<i>expr</i>)</code>	soma nós em <i>expr</i>	<code>sum(//cerveja/@preco)</code>
<code>round(<i>n</i>)</code>	arredonda <i>n</i>	<code>round(//cerveja/@preco)</code>

O detalhe e lista completa das funções XPath (2.0) está disponível em
<http://www.w3.org/TR/xpath-functions/>, ou
http://www.w3schools.com/xpath/xpath_functions.asp

* Os operadores booleanos consideram TRUE: uma string com pelo menos um carácter, um número diferente de zero, um ou mais nós XPath

O que o XPath fornece e o que não fornece...

■ O XPath permite

- ❑ tratar um documento uniformemente como uma árvore de nós
- ❑ especificar caminhos que percorrem a estrutura de um documento
- ❑ obter os nós encontrados no percurso de um documento
- ❑ aplicar condições a serem satisfeitas num determinado caminho

■ O XPath não permite

- ❑ combinar os nós obtidos em diferente caminhos
- ❑ definir caminhos que seguem as referências de atributos do tipo `IDREF`
- ❑ especificar relações de ordem entre os nós de uma lista
- ❑ expressões com quantificadores (e.g. “existe pelo menos um nó que...”)
- ❑ operações de união intersecção e diferença entre listas de nós

Qual o próximo passo?

- A capacidade declarativa do XPath é explorada em várias vertentes
 - pelo que “ganhar à vontade” com XPath é requisito para avançar!
- O XSLT (“eXtensible Stylesheet Language Transformations”)
 - usa o XPath para encontrar informação num documento XML
- O XSD (“XML Schema Definition”)
 - usa o XPath para especificar restrições à estrutura de um documento
- O XQuery e o XPath seguem o mesmo modelo
 - o XQuery assume os mesmos operadores e funções que o XPath
 - ... e adiciona construtores que eliminam as limitações do Path
 - ... “o XQuery é para o XML o que SQL é para o modelo relacional”

Um exemplo de sistema de gestão de bases de dados “open source” com suporte nativo para XML e que responde a XQuery está disponível em:

`http://exist.sourceforge.net/`

... sobre o bar “Pinguim”!



“Na origem da Antártica”
Ribeirão Preto, Brasil